## **DATA Segment:**

This segment declares the data used in the program.

BLOCK1 DB 'MALAYALAM'

* This line declares a byte array called BLOCK1 and initializes it with the string "MALAYALAM". This is the string that will be checked for palindrome.

MSG1 DB "IT IS PALINDROME $"

* This line declares a string message called MSG1 that says "IT IS PALINDROME" and ends with a "$" character.

MSG2 DB "IT IS NOT PALINDROME $"

* This line declares a string message called MSG2 that says "IT IS NOT PALINDROME" and ends with a "$" character.

PAL DB 00H

* This line declares a byte variable called PAL and initializes it with the value 00H. This variable is used later in the code to store a flag value.

## **PRINT Macro:**

This macro defines a subroutine for printing messages to the screen.

MOV AH,09H

* This line sets the value of the AH register to 09H, which is the code for the "print string" function in DOS.

LEA DX,MSG

* This line loads the address of the message to be printed into the DX register.

INT 21H

* This line calls the DOS interrupt 21H, which prints the message stored in the DX register to the screen.

INT 3H

* This line calls the software interrupt 3H, which terminates the program.

## **EXTRA Segment:**

This segment declares an extra block of memory called BLOCK2.

BLOCK2 DB 9 DUP(?)

* This line declares a byte array called BLOCK2 with 9 elements and initializes each element to an unspecified value with the "?" character.

## **CODE Segment:**

This segment contains the main logic of the program.

ASSUME CS:CODE,DS:DATA,ES:EXTRA

* This line tells the assembler which segments are being used in the program.

START:

* This line marks the beginning of the program.

MOV AX,DATA

* This line moves the address of the DATA segment into the AX register.

MOV DS,AX

* This line sets the DS register to the address of the DATA segment.

MOV AX,EXTRA

* This line moves the address of the EXTRA segment into the AX register.

MOV ES,AX

* This line sets the ES register to the address of the EXTRA segment.

LEA SI,BLOCK1

* This line loads the address of BLOCK1 into the SI register.

LEA DI,BLOCK2+8

* This line loads the address of the last element of BLOCK2 into the DI register. This will be the starting point for copying the BLOCK1 string to BLOCK2.

MOV CX,00009H

* This line sets the CX register to the value 00009H, which is the number of characters in the BLOCK1 string.

BACK:

* This line marks the beginning of a loop that will copy the BLOCK1 string to BLOCK2 in reverse order.

CLD

* This line clears the direction flag, which ensures that the string will be copied forward (i.e. from SI to DI).

LODSB

* This line loads the byte at the address pointed to by SI into the AL register.

STD

* This line sets the direction flag, which ensures that the string will be copied backwards (i.e. from DI to SI).

STOSB

* This line stores the byte in the AL register at the address pointed to by DI.

LOOP BACK

* This line decrements the CX register and jumps back to the